Efficient Social Media Post Content Generation Using Natural Language Processing Models

Adarsh Rawat[1]

Abstract

There are often times when people get into a loop while writing long, more expressive captions or content for posts. Natural Language Processing (NLP) and language models have gained high attention in the past few years to handle such real-life problems. A personalized AI-based platform is a better solution for it. Text generation is an essential use case of most models and can be utilized to handle it more effectively and efficiently by fine-tuning the model on user profile data. In our approach, we utilized techniques such as tokenization, embeddings, and sequence generation with the GPT-2 pre-trained model to generate responses. Challenges while implementing the solution are related to cost and computation; even only the fine-tuning step requires more computation. When put into a real-world use case, it will need continuous fine-tuning. Every time a user interacts with this platform, it will ask for fine-tuning, which means the model needs to be live on a cloud endpoint every time.

Keywords - Natural Language Processing, Large Language Models, Generative pretrained transformer

I. Introduction

Social media platforms have always been an integral part of our lives. The reason is the engaging content posted and made available there, which keeps users captivated for extended periods. Text content is a popular medium for this purpose. However, selecting the right text content, or crafting the best caption for a post, poses a challenge. It involves gathering data and knowledge and then reframing it for the intended audience. Even after this effort, maintaining a regular post flow may be difficult, and the entire process may require anywhere from half an hour to an hour.

Natural Language Processing (NLP) technologies and models are rapidly gaining recognition in today's tech world. Tools like ChatGPT, Rix, and Bard have become popular in various services. These models are used for content generation through prompt texts and can also be used to generate post content. However, we must also consider the specific user data to ensure that the generated output maintains post flow and relevance.

Our approach is to use a pretrained NLP model fine-tuned with user profile data. This allows for more specific generated output. Pretrained models can be obtained from the popular Hugging Face platform, which provides an AI-engaged community. Then, their 'auto-transformer' tool can be used for fine-tuning the model.

For fine-tuning, data can be obtained by using the previous posts and then input into the pretrained model. After fine-tuning, we can prompt the model to generate post content that is better suited to the user profile.

II. Literature Review

There exists a lot of time-efficient and optimal solutions built with Large Language Models following the advent of GPT (Generative Pretrained Transformer) technologies and NLP systems. Structured text prompting is a basic approach for post-content generation. Consequently, numerous researchers have investigated and explored diverse solution methodologies and solutions built upon this foundation. Their research provides insightful findings that can be utilized in research applications:

1. Li, C., & Xing, W. (2021).

This research explores the utilization of advanced deep learning models, particularly Recurrent Neural Networks (RNNs) and GPT-2, to deliver more human-like responses in Massive Open Online Courses (MOOCs) discussion forums. The researchers trained both RNN and GPT-2 models using an extensive dataset of post-replies from MOOC discussion forums and subsequently evaluated their performance by analyzing factors such as word readability and coherence. The results favor GPT-2 in terms of generating informational, emotional, and community support responses.

2. Hirschberg, J., & Manning, C. D. (2015).

This study delves into the field of Natural Language Processing (NLP) and its evolution over time. Initially, NLP research focused on defining human language rules and vocabularies for computers. During the 1990s, NLP began employing statistical methods and large datasets to enhance language comprehension. Modern NLP systems utilize complex machine learning models and possess a deep understanding of language structure. A crucial challenge in NLP lies in the limited resources and systems available for low-resource languages (languages lacking the necessary NLP tools and datasets).

3. Coppersmith, G., Leary, R., Crutchley, P., & Fine, A. (2018).

This paper highlights the application of social media data for detecting and assessing suicide risk. Researchers gathered digital life data through their portal OurDataHelps.org and employed deep learning techniques involving word embeddings, bidirectional Long Short-Term Memory (LSTM) layers, self-attention mechanisms, and linear layers of text classification to analyze language used in social media posts. Performance is measured through Receiver Operating Characteristic (ROC) curves, and the obtained results yield ROC curve Area Under the Curve (AUC) values ranging from 0.7 to 0.85.

4. Nijhawan, T., Attigeri, G., & Ananthakrishna, T. (2022).

This research focuses on leveraging social media data to analyze and predict an individual's emotional state. The paper mentions web scraping and tracking specific hashtags on social platforms for data collection. Pre-trained Language models like BERT were used for sentiment analysis, specifically a 5-class emotion classification, and Latent Dirichlet Allocation (LDA) was employed for identifying topics from documents.

5. Jahan, M. S., & Oussalah, M. (2023).

This study introduces an automated textual hate speech detection model utilizing deep learning and NLP techniques. A total of 69 datasets in 21 different languages were identified, primarily sourced from social platforms, with Twitter accounting for 45% of the total datasets. The paper references some top-ranked open-source projects that utilize various models and datasets for hate speech detection, including the use of TF-IDF, n-gram features, LR-SVC model architecture, Fast Text, CNN, and LSTM models.

III. Proposed System

The proposed system combined various different branches that are crucial for its implementation. As mentioned, we are discussing a model that can generate captions so efficiently that prompting is no longer necessary. This sets it apart from existing systems that rely on general prompting.

Here are the key elements of the system:

1. Natural Language Processing (NLP)

NLP is one of the most popular buzzwords in the AI world, and for good reason. The applications and models it have produced are so sophisticated that it can be difficult to distinguish between real and AI-generated text. By definition, NLP is the process of building machines or models that can understand (process) and manipulate human languages. Initially, NLP was developed to help computers understand the principles of languages. There are two main subparts of NLP: Natural Language Understanding (NLU), which focuses on semantic analysis or finding the meaning of text, and Natural Language Generation (NLG), the second step of the NLP process, which generates text by a machine. NLP models work in a similar way to ML models to find similarities between parts of languages, such as letters and words. They use different methods for tasks such as data preprocessing, feature extraction, and modeling. Some of the traditional machine learning and deep learning techniques include logistic regression, Naive Bayes, Latent Dirichlet Allocation (LDA), Recurrent Neural Network (RNN), Auto-Encoders, Transformers, etc.

2. Pretrained Large Language Models (LLMs)

LLMs are an evolved version of traditional NLP models in terms of parameters, performance, reliability, and efficiency for tasks such as generating open-ended questions, responses, summarized texts, as well as code and content generation. The name "large language models" indicates that they are trained on massive datasets for a better understanding of human language.

According to LLM architecture, it typically has three elements:

a. Encoder - Text tokens cannot be directly used for deep learning techniques. An encoder converts these tokens into embeddings that place similar words together in a vector space, allowing LLM models to utilize them.

b. Attention Mechanism - This part directs the model's focus on the more important words or embeddings, and it is not separate from the encoder and decoder.

c. Decoder - It is the reverse operation of the encoder, converting processed output embeddings into tokens that can be understood by normal users.

3. Fine-tuning

Fine-tuning is an optimization or customization technique that helps to train a model for a specific task. In the real world, every AI service works differently. For example, a chatbot application for a medical diagnosis system will serve differently than a chatbot for a restaurant system. Both provide reliable communication with customers, but they are trained differently for specific purposes. There are various ways to fine-tune a model using deep learning frameworks such as Transformers trainers, TensorFlow Karas, Torch, etc.

In our proposed system, we ensure that it becomes more and more user-centric, which is why fine-tuning is important. However, the data poses a challenge. If we only choose LinkedIn as a platform, extracting all the data from a profile raises privacy concerns and can be cost-intensive. To address this, we took a simpler approach where the user provides five relevant posts to the model, and the model then trains on those posts. Then, tokenization of the post data takes place, and for LLM understanding, it will be converted into vector space in the Embedding Layer. Since we cannot directly have input-output, it makes no sense for a user to first write an appropriate prompt for each post as their input. That's why we are using sequence generation, which is used to guess the next appropriate sequence token based on past tokens and padding sequence to truncate those sequences so that every sequence has a similar length.

IV. WORKING MODEL

There exist numerous time-efficient and optimal solutions built with Large Language Models following the advent of GPT (Generative Pretrained Transformer) technologies and NLP systems. Structured text prompting is a fundamental approach for post-content generation. Consequently, numerous researchers have investigated and explored diverse solution methodologies and solutions built upon this foundation. Their research provides insightful findings that can be utilized in research applications.

User-centricity is a key aspect of the model, and therefore, user interface integration is essential. A web platform will seamlessly provide users with a way to train the model on past post content and then generate new content by prompting the model.

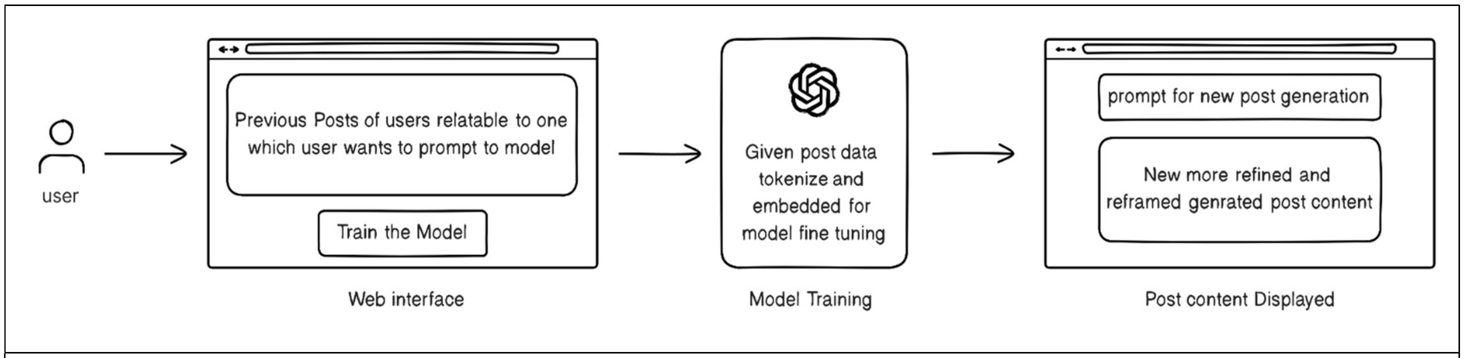
user
Previous Posts of users relatable to one which user wants to prompt to model
Train the Model
Web interface
Given post data tokenize and embedded for model fine tuning
Model Training
prompt for new post generation
New more refined and reframed genrated post content
Post content Displayed

Fig 1.1 Overview of flow and working of platform

The first interface will consist of a section where users will enter the past post content, they deem most suitable for generating new posts. In this step, the goal is for the model to pick up on and learn the flow used in these posts and how they would respond if there were a prompt. Fine-tuning is the most appropriate approach for this case, as it helps the model learn how to generate the next token by identifying associations with the previous one. The interface will provide four fields where users can input post content to train the model. Once the user clicks on "Train the model," the fine-tuning process will commence. There are two options for implementing the model: utilizing the OpenAI GPT-3 API or employing a pretrained GPT-2 model from Hugging Face. However, the fine-tuning of the GPT-3 API is a cost-intensive endeavor, and the GPT-2 model requires more computational power than a typical laptop system. Therefore, the pretrained GPT-2 fine-tuning model was chosen. The fine-tuning process itself is a comprehensive procedure that involves multiple steps and components, from tokenizing the text content to training the model on those tokens. This process is discussed in detail in the Project Architecture section. After successful model training, the generation of desired post content can commence. This step takes place in the second interface, where users prompt the model with specifications for the desired post, such as details about the topic, content length, and mood. The model will then generate content for the user based on the flow it learned during the training process.

This is how the entire working model functions to produce the final outcome. However, there are some limitations to consider. The model is either costly or computationally intensive, and due to limited resources for building the system, users can only submit a maximum of 5 posts. This decision can also impact the generated output, as even with fine-tuning, the model requires more and more post data to accurately identify and learn the flow and writing style.

V. PROJECT ARCHITECTURE

The section involves steps and methods from the very first step of getting data for fine-tuning until generating better, well-suited content from that data. Large Language Models (LLM) work with very high computation power on text data to obtain effective understanding, so more refined responses can be generated. But for this efficient outcome, there are some techniques and processes that help drive a model.

In our approach, we find the need for some of these processes to drive the model to a result, and hence here are their details:

1. Tokenization of Data

A complete sentence is made up of several words, and each word is made up of letters. Similarly, we receive a bunch of text as input for the model, but it's not possible for the model to process that whole text at once. So, the received text is broken into small fragments called tokens, which are then passed to the model. This process is known as tokenization. As per definition, it is the process of fragmenting the input and output texts into smaller units that can be processed by the LLM model. The type of token depends on the model size and its capability to handle those tokens. For example, if the model has enough computational backend to handle more granular, large-volume data, it is okay to generate characters and symbols as tokens; otherwise, words and formats are also a great choice.

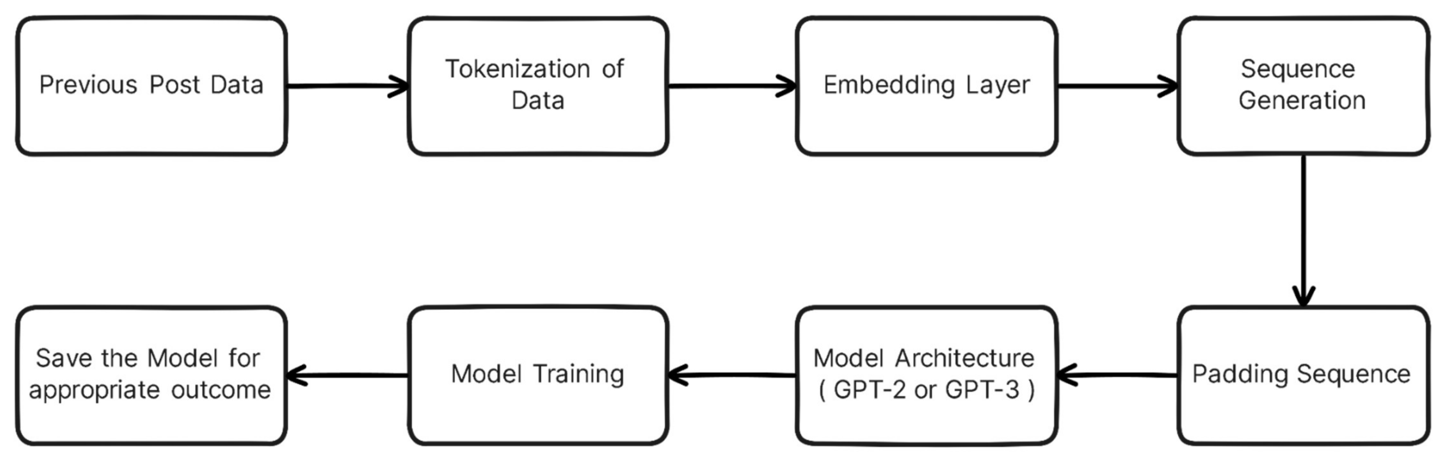

Fig 1.2 Various Important steps of Project Architecture

2. Embedding Layer

Now we have tokens of data, which are data chunks for the model to process the content more reliably and easily, aiding in accurate response generation. However, these small fragments of post content or data cannot be directly input to the model, as the model will not understand it. The model receives data in a specific format generally called tensors; the embedding layer transforms these chunks of data into a vector of numbers. This layer transforms these chunks of data into embeddings. Embedding allows models to understand the meaning of content in a more sophisticated way than traditional models, such as shot detection or keyframe extraction.

3. Sequence Generation and Padding Sequence

Sequence Generation is the method to generate a sequence of text in an NLP model. For example, if the content is "The cat is," the next probable word could be "sleeping," achieving a coherent and meaningful sentence. In our use case, one post content may not have a specific output to be labeled, thus predicting the next word as per the past ones could be a better way to do so. Thus, we use sequence generation methodology so during model fine-tuning, the model can learn how the content should be written. Now, after generating sequences, the padding of those is necessary so all the generated sequences can have the same length. This trains the model to generate content with the same length for each sequence.

4. Model Architecture

Choosing a model depends on several factors like its architecture and generated content. In terms of generating natural languages, GPT-models and transformer architecture work very well. A GPT model involves transformer architecture, which uses methods we discussed earlier, such as sequence generation and padding sequence. Also, for our system, we are suggested to use GPT-2, which is trained over 1.5 billion parameters. The goal of the GPT-2 model is to predict the next word; it can take past key values as input, which are the previously generated responses.

5. Model Training

GPT-2 model works on a guessing-next-word policy, which we discussed in the sequence generation section. Here we will not be training the model from scratch; it's a fine-tuning step. The used model is pre-trained; thus, it knows how natural language should be generated, which most appropriate word can lead that sentence, etc. Now, the data we have is the same as text-based data; the model will learn the flow of it and then, on prompting, will generate an output similar to the data content used in fine-tuning.

VI. Algorithms

The system employs techniques such as tokenization, embedding, sequence generation, and padding sequences. Although these methods are integral to the system's architecture, their implementation involves the use of algorithms. The project is closely aligned with the GPT-2 architecture, so the algorithms chosen need to be compatible and viable for the GPT-2 model. The following algorithm is used with the GPT-2 model architecture:

1. Byte-Pair Encoding (BPE) with GPT2-Tokenizer

BPE is a data compression technique used to tokenize text data into sub word tokens in NLP. The GPT2-Tokenizer utilizes BPE to create a unified vocabulary of sub word units. It begins with a single character and then proceeds to merge the most frequent pairs of units, forming a dynamic vocabulary that can represent both common and rare words within the content.

The Tokenizer, a part of the Hugging Face library, tokenizes the input post content into sub word units. It provides a vocabulary that includes whole words and sub word units, having been trained on a large dataset to help the model manage a wide range of linguistic choices.

2. Word Embeddings

Word embeddings represent a technique for converting words into numerical vectors, where words with similar meanings are assigned coordinates close to each other in a vector space. These embeddings capture the nuanced relationships and meanings of words based on their contextual patterns of usage. Notable methods for generating these embeddings include Word2Vec and Glove. Word2Vec relies on neural networks to predict the similarity of a word within a given text context, while Glove employs a global co-occurrence matrix. The resulting vectors portray words in a high-dimensional space, facilitating meaningful operations between them. This plays a pivotal role in augmenting our comprehension of words and optimizing the performance of models.

3. Transformer Architecture

Transformer represents a category of neural networks designed for comprehending natural languages through the analysis of sequential data. Leveraging mathematical techniques known as attention or self-attention mechanisms, transformers excel in recognizing the impact of distant data values and their interdependencies.

This architectural framework comprises an encoder-decoder structure reminiscent of Recurrent Neural Networks (RNNs) but distinguishes itself through the incorporation of attention mechanisms. The encoder encompasses N=6 functional layers, each featuring two sublayers: self-attention and a feed-forward network. Similarly, the decoder comprises N=6 functional layers, inclusive of sublayers, a decoder stack for past outputs, and a multi-head self-attention mechanism.

VII. CHALLENGES

There are two main challenges, common to working with language models: cost and computation power. Even fine-tuning a pre-trained language model requires significant computation. For fine-tuning a model, we often encounter two common but straightforward approaches. One is to download the pre-trained model and then fine-tune it, which requires high computation. The other is to use the OpenAI endpoint, as OpenAI provides model fine-tuning on their computational power but at a high cost. We discussed several approaches and techniques, from tokenization to sequencing and more. Each of these techniques takes data and performs processes on it, which is less intensive than actual training but, in parallel and continuous processing on data, incurs higher costs. If we consider the computational approach, it requires high-end CPU and memory resources to serve large datasets.

VIII. CONCLUSION

Personalized responses and agents represent the next great use case for language models. In the recent OpenAI Dev Day, they introduced GPTs, which are AI agents personalized for individuals. Personalized post content generation is a great help for those who spend hours collecting and reframing their posts. Currently, we inquire about past relatable posts to the user, but the process can become more accurate and streamlined by using social platform APIs. However, it requires additional access because one social platform will not directly allow third parties to use someone's profile data. After getting the data from the platform, we have to process it thoroughly before passing it to the model. A better approach, I assume, is to use APIs, efficiently handle and process data, and then pass it to the OpenAI endpoint for fine-tuning. The areas of use cases for language models are extensive, and the way things are developing, personalized platforms and agents are the future.

REFERENCES

[1] Li, C., & Xing, W. (2021). Natural language generation using deep learning to support MOOC learners. *International Journal of Artificial Intelligence in Education*, *31*, 186-214.

[2] Hirschberg, J., & Manning, C. D. (2015). Advances in natural language processing. *Science*, *349*(6245), 261-266.

[3] Coppersmith, G., Leary, R., Crutchley, P., & Fine, A. (2018). Natural language processing of social media as screening for suicide risk. *Biomedical informatics insights*, *10*, 1178222618792860.

[4] Nijhawan, T., Attigeri, G., & Ananthakrishna, T. (2022). Stress detection using natural language processing and machine learning over social interactions. *Journal of Big Data*, *9*(1), 1-24.

[5] Jahan, M. S., & Oussalah, M. (2023). A systematic review of Hate Speech automatic detection using Natural Language Processing. *Neurocomputing*, 126232.